18 | HTAP是不是赢者通吃的游戏？

2020-09-18 王磊

分布式数据库30讲

进入课程 >

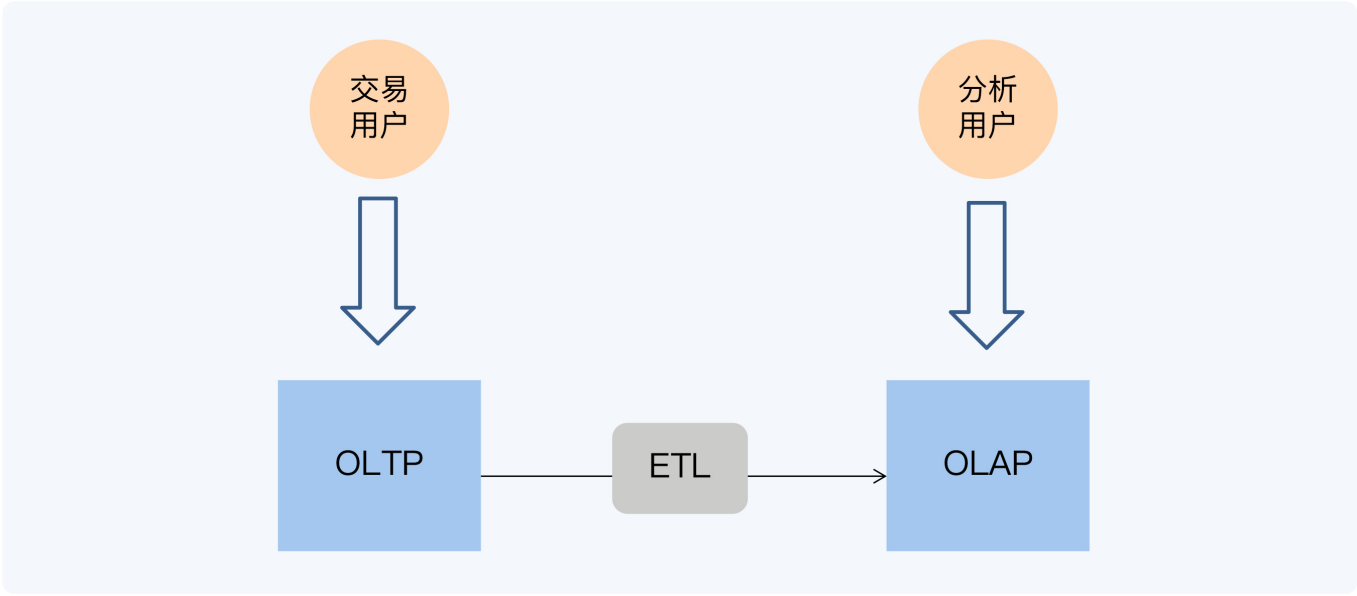
讲述：王磊

时长 18:26　大小 16.89M

你好，我是王磊，你也可以叫我 Ivan。

这一讲的关键词是 HTAP，在解释这个概念前，我们先要搞清楚它到底能解决什么问题。

有关 OLTP 和 OLAP 的概念，我们在第 1 讲就已经介绍过了。OLTP 是面向交易的处理过程，单笔交易的数据量很小，但是要在很短的时间内给出结果；而 OLAP 场景通常是基于大数据集的运算。



OLAP 和 OLTP 通过 ETL 进行衔接。为了提升 OLAP 的性能，需要在 ETL 过程中进行大量的预计算，包括数据结构的调整和业务逻辑处理。这样的好处是可以控制 OLAP 的访问延迟，提升用户体验。但是，因为要避免抽取数据对 OLTP 系统造成影响，所以必须在日终的交易低谷期才能启动 ETL 过程。这样一来， OLAP 与 OLTP 的数据延迟通常就在一天左右，习惯上大家把这种时效性表述为 T+1。其中，T 日就是指 OLTP 系统产生数据的日期，T+1 日是 OLAP 中数据可用的日期，两者间隔为 1 天。

你可能已经发现了，这个体系的主要问题就是 OLAP 系统的数据时效性，T+1 太慢了。是的，进入大数据时代后，商业决策更加注重数据的支撑，而且数据分析也不断向一线操作渗透，这都要求 OLAP 系统更快速地反映业务的变化。

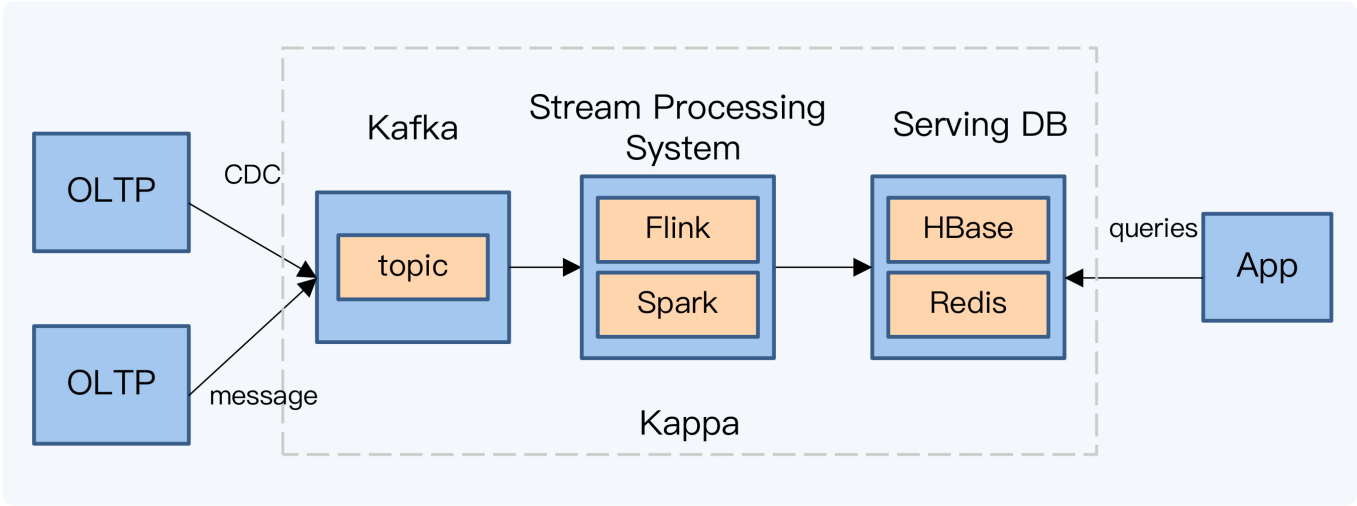
两种解决思路

说到这，你应该猜到了，HTAP 要解决的就是 OLAP 的时效性问题，不过它也不是唯一的选择，这个问题有两种解决思路：

1. 用准实时数据计算替代原有批量 ETL 过程，重建 OLAP 体系；
2. 弱化甚至是干脆拿掉 OLAP，直接在 OLTP 系统内扩展，也就是 HTAP。

重建 OLAP 体系

我们先来看第一种思路。重建 OLAP 体系，重视数据加工的时效性，正是近年来大数据技术的主要发展方向。Kappa 架构就是新体系的代表，它最早由 LinkedIn 的 Jay Kreps 在 2014 年的一篇文章中提出。



在 Kappa 架构中，原来的批量文件传输方式完全被 Kafka 替代，通过流计算系统完成数据的快速加工，数据最终落地到 Serving DB 中提供查询服务。这里的 Serving DB 泛指各种类型的存储，可以是 HBase、Redis 或者 MySQL。

要注意的是，Kappa 架构还没有完全实现，因为在实践中流计算仍然无法替代批量计算，Serving DB 也无法满足各种类型的分析查询需求。未来，Kappa 架构需要在两方面继续完善：

1. 流计算能力的增强，这需要用到 Kafka 和 Flink 等软件；
2. Serving DB 即时计算能力的增强，这就寄希望于 OLAP 数据库的突破，就像 ClickHouse 已经做的那样。

总的来说，新的 OLAP 体系试图提升即时运算能力，去除批量 ETL，降低数据延迟。这个新体系是流计算的机遇，也是 OLAP 数据库的自我救赎。

新建 HTAP 系统

第二种思路是 HTAP。HTAP（Hybrid Transaction/Analytical Processing）就是混合事务分析处理，它最早出现在 2014 年 Gartner 的一份报告中，很巧和 Kappa 架构是同一年。Gartner 用 HTAP 来描述一种新型数据库，它打破了 OLTP 和 OLAP 之间的隔阂，在一个数据库系统中同时支持事务型数据库场景和分析型数据库场景。这个构想非常美妙，HTAP 可以省去繁琐的 ETL 操作，避免批量处理造成的滞后，更快地对最新数据进行分析。

这个构想很快表现出它侵略性的一面，由于数据产生的源头在 OLTP 系统，所以 HTAP 概念很快成为 OLTP 数据库，尤其是 NewSQL 风格的分布式数据库，向 OLAP 领域进军的一面旗帜。

那么，NewSQL 在初步解决 OLTP 场景的高并发、强一致性等问题后，能不能兼顾 OLAP 场景，形成赢者通吃的局面呢？

其实还很难讲，因为从技术实践看，重建 OLAP 路线的相关技术似乎发展得更快，参与厂商也更加广泛，在实际生产环境的落地效果也不断改善。

相比之下，HTAP 的进展比较缓慢，鲜有生产级的工业实践，但仍有不少厂商将其作为产品的演进方向。目前，厂商官宣的 HTAP 至少包括 TiDB 和 TBase, 而 OceanBase 也宣布在近期版本中推出 OLAP 场景的特性。基于商业策略的考虑，我相信未来还会有更多分布式数据库竖起 HTAP 的大旗。那么接下来，我们分析下 HTAP 面临的挑战，让你更好地识别什么是 HTAP。

HTAP 的两种存储设计

这就要先说回 OLTP 和 OLAP，在架构上，它们的差异在于计算和存储两方面。

计算是指计算引擎的差异，目标都是调度多节点的计算资源，做到最大程度地并行处理。因为 OLAP 是海量数据要追求高吞吐量，而 OLTP 是少量数据更重视低延迟，所以它们计算引擎的侧重点不同。

存储是指数据在磁盘上的组织方式不同，而组织方式直接决定了数据的访问效率。OLTP 和 OLAP 的存储格式分别为行式存储和列式存储，它们的区别我稍后会详细说明。

分布式数据库的主流设计理念是计算与存储分离，那么计算就比较容易实现无状态化，所以在一个 HTAP 系统内构建多个计算引擎显然不是太困难的事情，而真的要将 HTAP 概念落地为可运行系统，根本性的挑战就是存储。面对这个挑战，业界有两个不同的解决思路：

1. Spanner 使用的融合性存储 PAX（Partition Attributes Across），试图同时兼容两类场景。

2. TiDB4.0 版本中的设计，在原有行式存储的基础上，新增列式存储，并通过创新性的设计，保证两者的一致性。

Spanner：存储合一

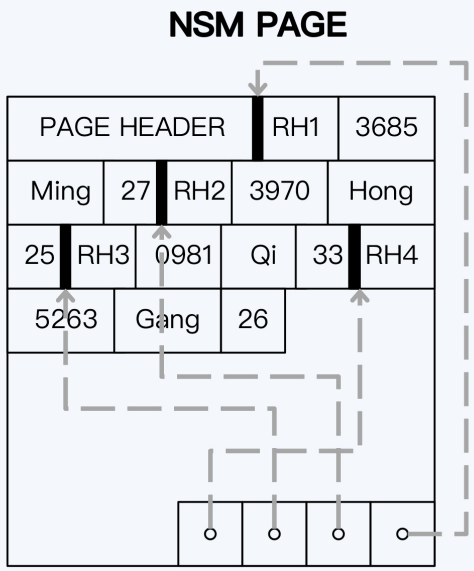
首先，我们一起看看 Spanner 的方案。Spanner2017 论文“Spanner: Becoming a SQL System”中介绍了它的新一代存储 Ressi，其中使用了类似 PAX 的方式。这个 PAX 并不是 Spanner 的创新，早在 VLDB2002 的论文 “Data Page Layouts for Relational Databases on Deep Memory Hierarchies” 中就被提出了。论文从 CPU 缓存友好性的角度，对不同的存储方式进行了探讨，涉及 NSM、DSM、PAX 三种存储格式。

NSM （行式存储）

NSM（N-ary Storage Model）就是行式存储，也是 OLTP 数据库默认的存储方式，始终伴随着关系型数据库的发展。我们常用的 OLTP 数据库，比如 MySQL（InnoDB）、PostgreSQL、Oracle 和 SQL Server 等等都使用了行式存储。

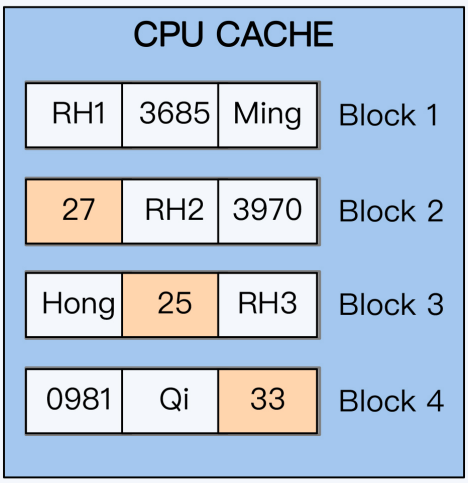
顾名思义，行式存储的特点是将一条数据记录集中存在一起，这种方式更加贴近于关系模型。写入的效率较高，在读取时也可以快速获得一个完整数据记录，这种特点称为记录内的局部性（Intra-Record Spatial Locality）。

RID	SSN	Name	Age
1	3865	Ming	27
2	3970	Hong	25
3	0981	Qi	33
4	5263	Gang	26



但是，行式存储对于 OLAP 分析查询并不友好。OLAP 系统的数据往往是从多个 OLTP 系统中汇合而来，单表可能就有上百个字段。而用户一次查询通常只访问其中的少量字段，如果以行为单位读取数据，查询出的多数字段其实是无用的，也就是说大量 I/O 操作都是

无效的。同时，大量无效数据的读取，又会造成 CPU 缓存的失效，进一步降低了系统的性能。



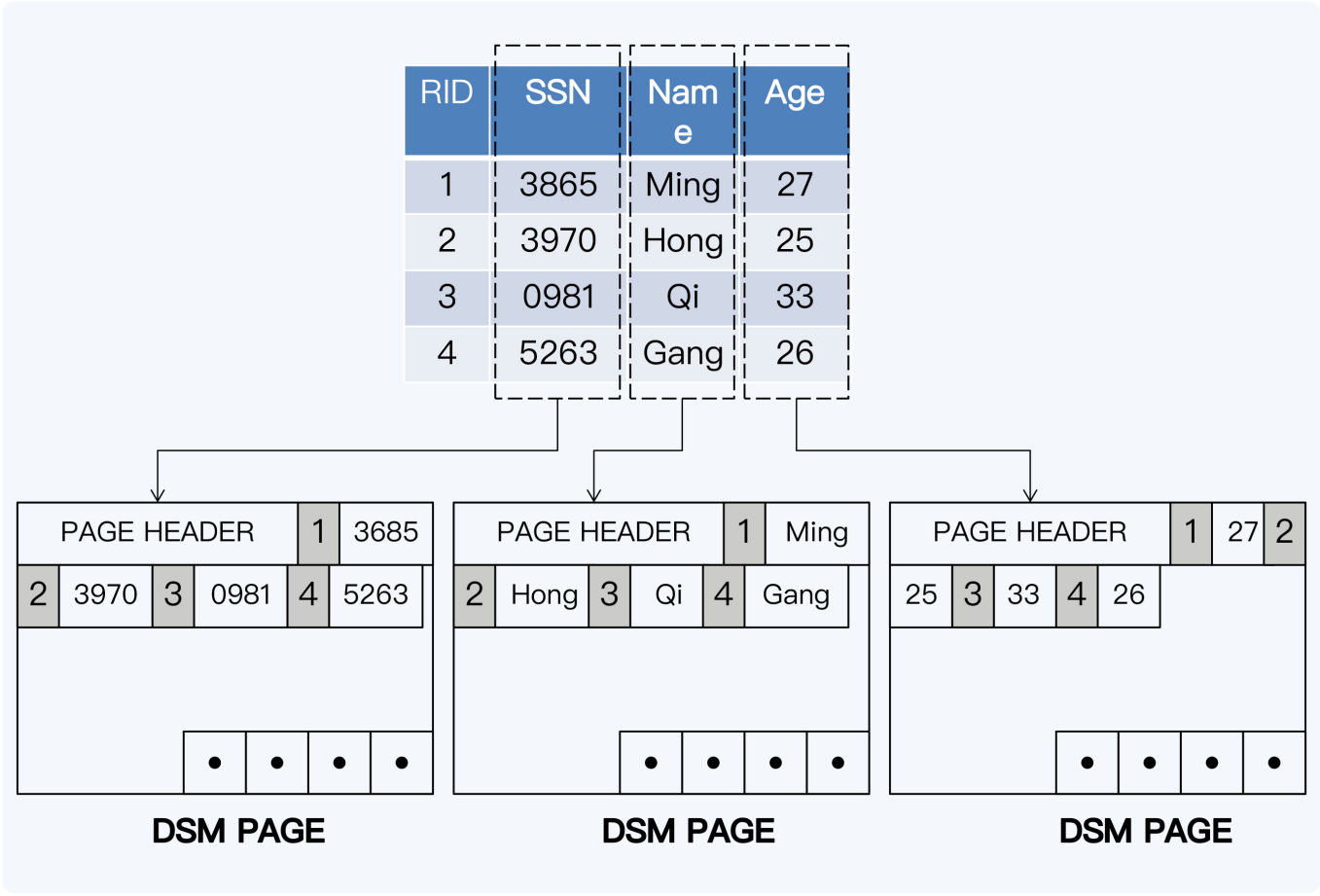
图中显示 CPU 缓存的处理情况，我们可以看到很多无效数据被填充到缓存中，挤掉了那些原本有机会复用的数据。

DSM（列式存储）

DSM（Decomposition Storage Model）就是列式存储，它的出现要晚于行式存储。典型代表系统是 C-Store，它是迈克尔 · 斯通布雷克（Micheal Stonebraker）主导的开源项目，后来的商业化产品就是 Vertica。

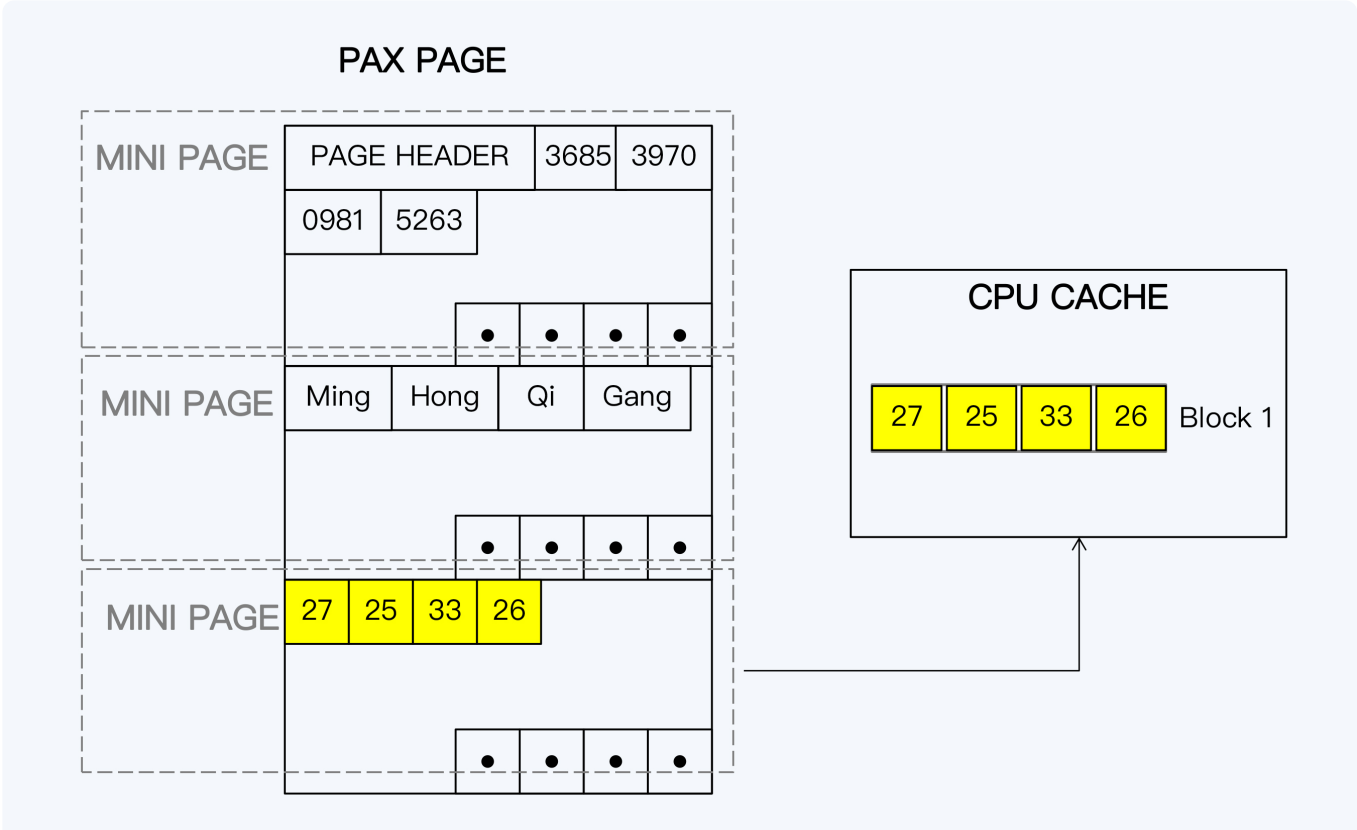
列式存储就是将所有列集中存储，不仅更加适应 OLAP 的访问特点，对 CACHE 也更友好。这种特点称为记录间的局部性（Inter-Record Spatial Locality）。列式存储能够大幅提升查询性能，以速度快著称的 ClickHouse 就采用了列式存储。

列式存储的问题是写入开销更大，这是因为根据关系模型，在逻辑上数据的组织单元仍然是行，改为列式存储后，同样的数据量会被写入到更多的数据页（page）中，而数据页直接对应着物理扇区，那么磁盘 I/O 的开销自然增大了。



列式存储的第二个问题，就是很难将不同列高效地关联起来。毕竟在多数应用场景中，不只是使用单列或单表数据，数据分散后，关联的成本会更高。

PAX



PAX 增加了 minipage 这个概念，是原有的数据页下的二级单位，这样一行数据记录在数据页上的基本分布不会被破坏，而相同列的数据又被集中地存储在一起。PAX 本质上还是更接近于行式存储，但它也在努力平衡记录内局部性和记录间局部性，提升了 OLAP 的性能。

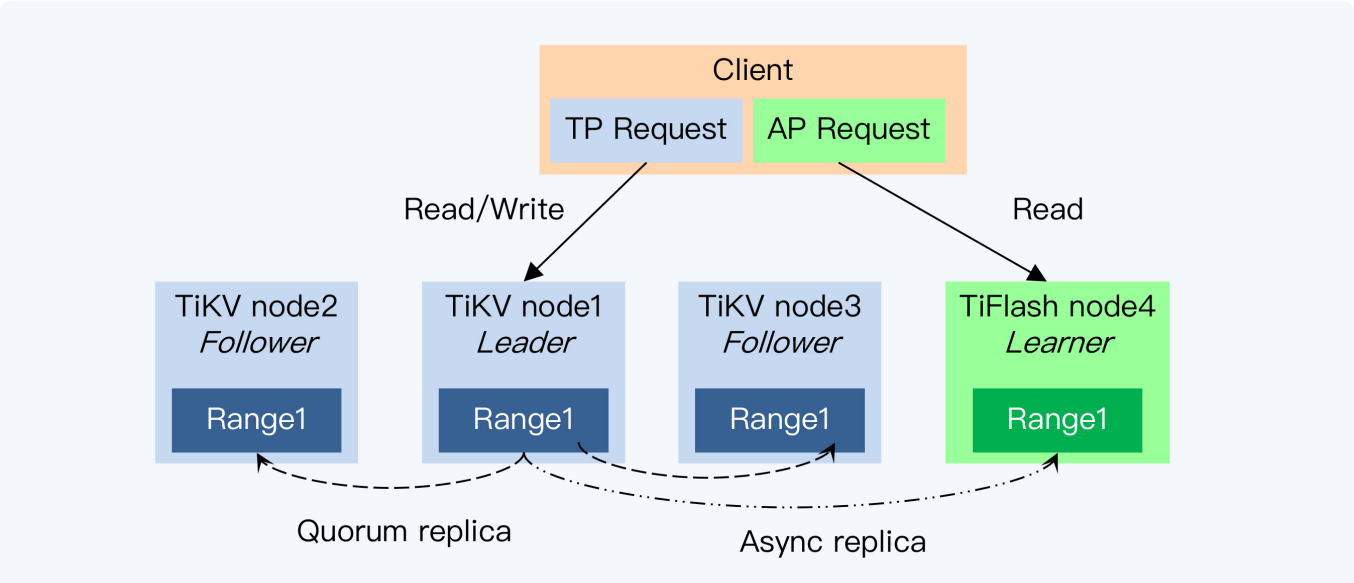
理论上，PAX 提供了一种兼容性更好的存储方式，可让人有些信心不足的是其早在 2002 年提出，但在 Spanner 之前却少有落地实现。

与这个思路类似的设计还有 HyPer 的 DataBlock(SIGMOD2016)，DataBlock 构造了一种独有的数据结构，同时面向 OLTP 和 OLAP 场景。

TiFlash：存储分离

如果底层存储是一份数据，那么天然就可以保证 OLTP 和 OLAP 的数据一致性，这是 PAX 的最大优势，但是由于访问模式不同，性能的相互影响似乎也是无法避免，只能尽力选择一个平衡点。TiDB 展现了一种不同的思路，介于 PAX 和传统 OLAP 体系之间，那就是 OLTP 和 OLAP 采用不同的存储方式，物理上是分离的，然后通过创新性的复制策略，保证两者的数据一致性。

TiDB 是在较早的版本中就提出了 HTAP 这个目标，并增加了 TiSpark 作为 OLAP 的计算引擎，但仍然共享 OLTP 的数据存储 TiKV，所以两种任务之间的资源竞争依旧不可避免。直到近期的 4.0 版本中，TiDB 正式推出了 TiFlash 作为 OLAP 的专用存储。



我们的关注点集中在 TiFlash 与 TiKV 之间的同步机制上。其实，这个同步机制仍然是基于 Raft 协议的。TiDB 在 Raft 协议原有的 Leader 和 Follower 上增加了一个角色 Learner。这个 Learner 和 Paxos 协议中的同名角色，有类似的职责，就是负责学习已经达成一致的状态，但不参与投票。这就是说，Raft Group 在写入过程中统计多数节点时，并没有包含 Learner，这样的好处是 Learner 不会拖慢写操作，但带来的问题是 Learner 的数据更新必然会落后于 Leader。

看到这里，你可能会问，这不就是一个异步复制吗，换了个马甲而已，有啥创新的。这也保证不了 AP 与 TP 之间的数据一致性吧？

Raft 协议能够实现数据一致性，是因为限制了只有主节点提供服务，否则别说是 Learner 就是 Follower 直接对外服务，都不能满足数据一致性。所以，这里还有另外一个设计。

Learner 每次接到请求后，首先要确认本地的数据是否足够新，而后才会执行查询操作。怎么确认足够新呢？ Learner 会拿着读事务的时间戳向 Leader 发起一次请求，获得 Leader 最新的 Commit Index，就是已提交日志的顺序编号。然后，就等待本地日志继续 Apply，直到本地的日志编号等于 Commit Index 后，数据就足够新了。而在本地 Region 副本完成同步前，请求会一直等待直到超时。

这里，你可能又会产生疑问。这种同步机制有效运转的前提是 TiFlash 不能落后太多，否则每次请求都会带来数据同步操作，大量请求就会超时，也就没法实际使用了。但是，TiFlash 是一个列式存储，列式存储的写入性能通常不好，TiFlash 怎么能够保持与 TiKV 接近的写入速度呢？

这就要说到 TiFlash 的存储引擎 Delta Tree，它参考了 B+ Tree 和 LSM-Tree 的设计，分为 Delta Layer 和 Stable Layer 两层，其中 Delta Layer 保证了写入具有较高的性能。因为目前还没有向你介绍过存储引擎的背景知识，所以这里不再展开 Delta Tree 的内容了，我会在第 22 讲再继续讨论这个话题。

当然，TiFlash 毕竟是 OLAP 系统，首要目标是保证读性能，因此写入无论多么重要，都要让位于读优化。作为分布式系统，还有最后一招可用，那就是通过扩容降低单点写入的压力。

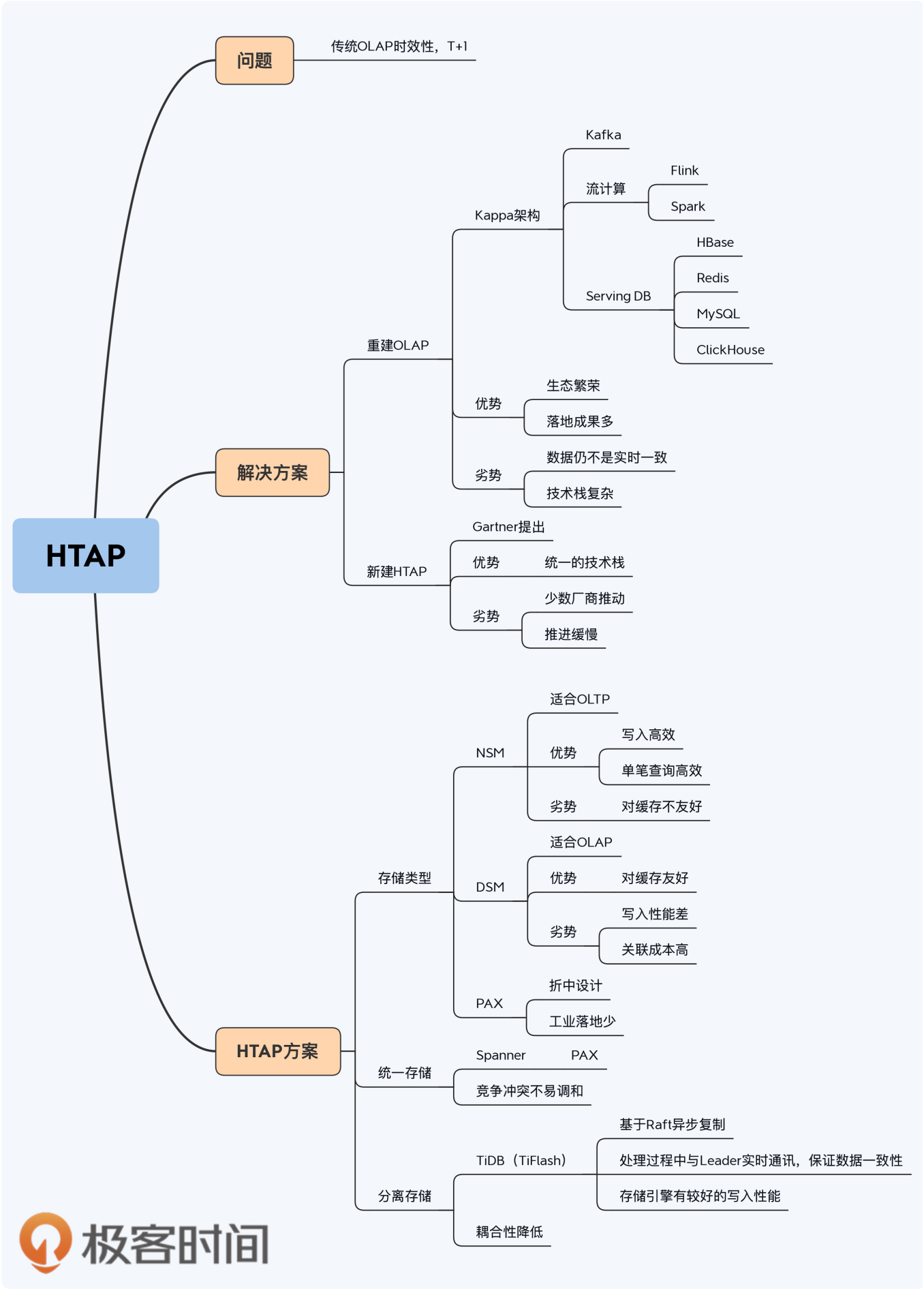
小结

好了，今天的内容我们就聊到这了，最后让我们梳理一下这一讲的要点。

1. OLTP 通过 ETL 与 OLAP 衔接，所以 OLAP 的数据时效性通常是 T+1，不能及时反映业务的变化。这个问题有两种解决思路，一种是重建 OLAP 体系，通过流计算方式替代批量数据处理，缩短 OLAP 的数据延迟，典型代表是 Kappa 架构。第二种思路是 Gartner 提出的 HTAP。
2. HTAP 的设计要点在计算引擎和存储引擎，其中存储引擎是基础。对于存储引擎也两种不同的方案，一种是以 PAX 为代表，用一份物理存储融合行式和列式的特点，Spanner 采用了这种方式。另一种是 TiDB 的 TiFlash，为 OLTP 和 OLAP 分别设置行式存储和列式存储，通过创新性的同步机制保证数据一致。
3. TiDB 的同步机制仍然是基于 Raft 协议的，通过增加 Learner 角色实现异步复制。异步复制必然带来数据的延迟，Learner 在响应请求前，通过与 Leader 同步增量日志的方式，满足数据一致性，但这会带来通讯上的额外开销。
4. TiFlash 作为列存，首先要保证读取性能，但因为要保证数据一致性，所以也要求具备较高的写入性能，TiFlash 通过 Delta Tree 的设计来平衡读写性能。这个问题我们没有展开，将在 22 讲继续讨论。

总的来说，HTAP 是解决传统 OLAP 的一种思路，但是推动者只是少数 OLTP 数据库厂商。再看另一边，以流计算为基础的新 OLAP 体系，这些技术本身就是大数据技术生态的一部分，有更多的参与者，不断有新的成果落地。至于 HTAP 具有绝对优势的数据一致性，其实在商业场景中也未必有足够多的刚性需求。所以，从实践效果出发，我个人更加看好后者，也就是新 OLAP 体系。

当然 HTAP 也具有相对优势，那就是通过全家桶方案避免了用户集成多个技术产品，整体技术复杂度有所降低。最后，TiDB 给出的解决方案很有新意，但是能否覆盖足够大的 OLAP 场景，仍有待观察。


HTAP
问题
传统OLAP时效性，T+1
解决方案
重建OLAP
Kappa架构
Kafka
流计算
Flink
Spark
Serving DB
HBase
Redis
MySQL
ClickHouse
优势
生态繁荣
落地成果多
劣势
数据仍不是实时一致
技术栈复杂
新建HTAP
Gartner提出
优势
统一的技术栈
劣势
少数厂商推动
推进缓慢
HTAP方案
存储类型
NSM
适合OLTP
优势
写入高效
单笔查询高效
劣势
对缓存不友好
DSM
适合OLAP
优势
对缓存友好
劣势
写入性能差
关联成本高
PAX
折中设计
工业落地少
统一存储
Spanner
PAX
竞争冲突不易调和
分离存储
TiDB（TiFlash）
基于Raft异步复制
处理过程中与Leader实时通讯，保证数据一致性
存储引擎有较好的写入性能
耦合性降低
极客时间

思考题

课程的最后，我们来看看今天的思考题。今天我们介绍了 TiDB 的 OLAP 组件 TiFlash，它保持数据一致性的方法是，每次 TiFlash 接到请求后，都会向 TiKV Leader 请求最新的日志增量，本地 replay 日志后再继续处理请求。这种模式虽然能够保证数据足够新，但比起 TiFlash 独立服务多了一次网络通讯，在延迟上有较大的影响。我的问题就是，你觉得这个模式还能优化吗？在什么情况下不需要与 Leader 通讯？

欢迎你在评论区留言和我一起讨论，我会在答疑篇回复这个问题。如果你身边的朋友也对 HTAP 这个话题感兴趣，你也可以把今天这一讲分享给他，我们一起讨论。

学习资料

Anastassia Ailamaki et al.: *Data Page Layouts for Relational Databases on Deep Memory Hierarchies*

Harald Lang et al: *Data Blocks: Hybrid OLTP and OLAP on Compressed Storage using both Vectorization and Compilation*

Jay Kreps: *Questioning the Lambda Architecture*

Nigel Rayner et al.: *Hybrid Transaction/Analytical Processing Will Foster Opportunities for Dramatic Business Innovation*

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 17 | 为什么不建议你使用自增主键?

下一篇 19 | 查询性能优化：计算与存储分离架构下有哪些优化思路?

精选留言 (4)

写留言

游弋云端
2020-09-18

可以后台启动一个轮询日志增量的线程，当差异大于一定量的时候触发实际的数据同步。或者在心跳包中增加一个版本用于比对，当差异大的时候，触发主动同步。这样不用等到请求到达时触发，省掉这个等待时延。但是由于是Raft的非成员节点，怎么做都会有一定的数据差异，单对于大多OLAP分析场景应该是足够使用了。

展开

2

tt
2020-09-18

没有接触过OLAP。

是不是可以不用每次都去请求“最新”的日志增量，而是按需请求数据：本地保存一个数据新旧的时间戳，如果早于读请求的时间戳，就不用去请求了；

...

展开

1

myrfy

2020-09-18

当客户请求的时间戳可以确信小于服务端的时间戳时。难点应该就是如何保证客户端和服务端在时间上的同步。

展开

Fan()

2020-09-18

受益匪浅

展开